RANKING FUNCTIONS FOR SIZE-CHANGE TERMINATION II

AMIR M. BEN-AMRAM [a] AND CHIN SOON LEE [b]

[a] School of Computer Science, The Tel-Aviv Academic College, Israel
 e-mail address: benamram.amir@gmail.com

[b] Singapore
 e-mail address: cslee@asteria.com.sg

ABSTRACT. Size-Change Termination is an increasingly-popular technique for verifying program termination. These termination proofs are deduced from an abstract representation of the program in the form of *size-change graphs*.

We present algorithms that, for certain classes of size-change graphs, deduce a *global ranking function*: an expression that ranks program states, and decreases on every transition. A ranking function serves as a witness for a termination proof, and is therefore interesting for program certification. The particular form of the ranking expressions that represent SCT termination proofs sheds light on the scope of the proof method. The complexity of the expressions is also interesting, both practicaly and theoretically.

While deducing ranking functions from size-change graphs has already been shown possible, the constructions in this paper are simpler and more transparent than previously known. They improve the upper bound on the size of the ranking expression from triply exponential down to singly exponential (for certain classes of instances). We claim that this result is, in some sense, optimal. To this end, we introduce a framework for lower bounds on the complexity of ranking expressions and prove exponential lower bounds.

1. INTRODUCTION

Automatic termination analysis is a rapidly growing field; it represents exciting progress in the theory and application of program analysis. Two widely-cited foundational publications are Podelski and Rybalchenko [11] and Lee, Jones and Ben-Amram [10]. The former promoted the use of *local ranking functions* (or relations) for termination proofs; the latter presented the Size-Change Termination (SCT) framework. SCT is, in essence, a class of abstract programs (in other words, transition systems), that happen to be conveniently represented as graphs (known as size-change graphs). These abstract programs can be used to safely approximate the semantics of an actual program, while capturing invariants that are crucial to a termination proof.

A *global ranking function* is a function of program states that decreases towards its lower bound in every program transition, thus providing a direct witness to termination. It is a folklore theorem that a program terminates if and only if such a function exists.

```
        while (x, y > 0)
          if x>y then x := x-y
              else y := y-x
        return max(x,y)
```

Figure 1: A gcd program. One ranking function for the loop is the maximum among the variables. Another is their sum.

But global ranking functions can be complex and difficult to find, even for simple kinds of programs (for example, the programs that we will consider in this work). Both [11] and [10] circumvent the construction of a global ranking function. In fact, they are closely related: as clearly explained in [6], SCT termination proofs fit the local framework of [11].

The problem addressed in this work is that of deriving an explicit expression for a global ranking function for a given SCT instance (despite the fact that a termination proof can be obtained without it). Why are global ranking functions interesting? Firstly, for theoretical understanding of size-change termination : identifying a class of ranking functions that captures all terminating instances provides a clue to the *scope* of the proof method (what behaviours are captured by the method, what ordinals may be captured etc.). Effectively constructing a ranking function is a challenge because it is not at all obvious how to do it, even when a function is known to exist; and developing such an algorithm provides new insights into the subject. Secondly, for applications: an explicit ranking expression may provide an easy-to-verify witness to termination, since verification only amounts to checking it against every transition. Such a witness is not provided by the local methods. As pointed out by Krauss [8], if a global ranking function (of sufficiently simple form) could be constructed automatically, it would allow a theorem prover to certify the termination claim while allowing the tool that searches for the termination proof to stay outside the trusted (formally verified) code base. One can also consider applications to proof-carrying code, where again the desire is for the proof to be given as a certificate that is easier to check than to find. Finally, an interesting potential application to bound the excution time of programs, since a ranking function provides a progress measure. However, such application is not immediate, since the range of our functions is, in general, not the integers (i.e., not the order type ω, but ω^k for some k).

For all these reasons, we are interested in the form and complexity of global ranking functions that suffice for SCT programs, and in algorithms for their automatic construction. Early publications on SCT identified several special cases: programs where the maximum, minimum, or sum of a set of variables decreases (Figure 1), programs with a lexicographic descent (the ubiquitous Ackermann's function), and programs with multiset-descent [13, 4]. Lee [9] established for the first time that a ranking function can be automatically constructed for *any terminating SCT instance*, always of the following form:

$$\min(\max S_1, \max S_2, \dots)$$

where $\max S_i$ represents the maximum element among a set S_i of *vectors* (tuples) of variables and constants, where vectors are lexicographically ordered.

Let us give an example. Figure 2 shows a program, and the graphs in Figure 3 represent it in SCT form. The three graphs correspond to the three alternatives in the loop body,

```
    while (x,y,z > 0)
       do either
         (x,y,z) := (y-1, y-1, f(x,y,z))
       | (x,y,z) := (x-1, x,   f(x,y,z))
       | (x,y,z) := (y-1, y,   z-1)
```

Figure 2: A terminating program. Function f is considered unknown. "do either" is nonde-
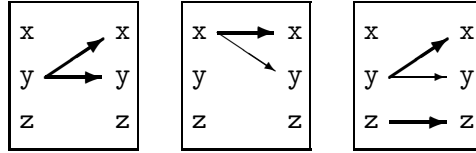terministic.

Figure 3: Size-change graphs for the program in Figure 2.

with arcs representing data flow and heavy arcs representing descent (for precise definitions
of size-change graphs, see Section 2).

A ranking function for this program is $\rho(x, y, z) = \max\{\langle y, 0, z\rangle, \langle x, 1, z\rangle\}$ (in this in-
stance the *min* operator is unnecessary; this could be expected, as Section 5 will show). A
straight-forward way to verify this is to check each graph in turn, considering each possibil-
ity among $y > x$, $y = x$ and $y < x$. For example, take the first graph. Assume that initially
$y > x$. Then the initial maximum is $\langle y, 0, z\rangle$; since the transition decreases y, there is no
doubt that ρ descends.

Contributions of this work. We provide new algorithms to construct ranking functions for
a restricted, but interesting, class of SCT instances: fan-in free (or fan-out free) graphs.
We feel that the new constructions are far more transparent than the previous one, which
involved a lengthy detour through the determinization of Büchi automata. In contrast,
the new ones are based on direct analysis of SCT graphs. They employ a technique of
including composite values, such as tuples or sets of variables, as single data items, and
showing that this simplifies the SCT analysis. The inclusion of tuples reflects the role of
lexicographic descent; sets of variables give rise to descent in *multiset orders* (see Section 2).
Thus the constructions also provide one more example of the usefulness of multiset orders
in termination-related reasoning. In terms of expression size, we reduce the upper bound
from triply-exponential as in [9] to singly-exponential.

An additional contribution of this work is the formulation of a lower-bound framework
and the proof of exponential lower bounds, which imply that our upper complexity bounds
are, in a certain sense, optimal.

Structure of this paper. The next section provides the formal introduction, giving necessary
definitions. In Section 3 we review some results that we are using from previous work.
Then, in Sections 4 and 5 we give the new construction for fan-out free, then fan-in free,

graphs. Section 6 is concerned with lower bounds. In Section 7, we discuss the connection of our results to a more general theorem of Blass and Gurevich. Section 8 concludes.

2. DEFINITIONS

In this section we list the necessary definitions involving SCT, ranking functions, and their connection. Enough definitions are given to make the paper formally self-contained and to fix the terminology (which is, unfortunately, not uniform across SCT-related work).

2.1. **Program representation.** Let *Val* be a well-ordered set of data values.

Definition 2.1. A control-flow graph (CFG) is a directed multigraph (F, C). The nodes are called flow-chart points or just flow-points. The set of arcs from $f \in F$ to $g \in F$ is denoted C_{fg}.

For each $f \in F$, we have a distinct set of *variables Var(f)*.

One of the nodes, f_0, is *initial* or starting point. All nodes are reachable from f_0.

Often, we further assume that f_0 is also reachable from all nodes, or equivalently that the graph is strongly connected; it is well-known that termination analysis can be done one strongly-connected component at a time.

The variables $Var(f)$ are supposed to represent data pertinent to the program state when the program is at point f.

To avoid cumbersome notations, we make in this paper the following assumption:

All sets $Var(f)$ have the same size n.

We also reserve the identifier m for $|F|$.

Definition 2.2. The set of (abstract) program states is

$$St = \{(f, \sigma) \mid f \in F, \ \sigma : Var(f) \to Val\} \,.$$

Thus, a state is defined by a program point and a *store* σ applicable to that point. A state will be cutomarily denoted by s and we sometimes implicitly assume that its components are (f, σ).

A remark about the notion of abstract state may be in order. While in simple settings (such as [10]), $Var(f)$ may correspond precisely to constituent variables of the concrete program state, this is not true in general. In many applications of SCT, static program analysis is used to determine properties of a state that are (or may be) relevant to termination, e.g., the difference of two integer variables, the depth of a recursion or closure stack, etc. The original presentation of SCT referred to programs that process data types, such as lists and trees, that can be ranked by their size, height etc; in this case we may prefer to think of the abstract value as a member of $Var(f)$ rather than the list or tree itself. There are also examples where it is worthwhile to include two abstractions (norms) of the same concrete object.

Definition 2.3. For $f, g \in F$, a size-change graph (SCG) with source f and target g is a bipartite directed graph with source nodes corresponding to $Var(f)$ and target nodes corresponding to $Var(g)$. We write this fact as $G : f \to g$. Each arc of G (called a *size-change arc*) is labeled with an element of the set $\{\downarrow, \overline{\downarrow}\}$.
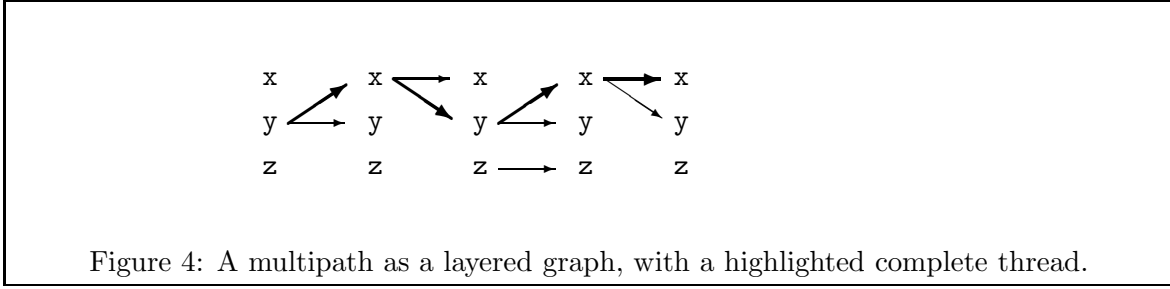
Figure 4: A multipath as a layered graph, with a highlighted complete thread.

Size-change arcs represent constraints on *state transitions* $(f, \sigma) \mapsto (g, \sigma')$. The arcs have the following meaning:

A *strict* arc $x \xrightarrow{\downarrow} y$ represents the assertion $\sigma(x) > \sigma'(y)$

A *non-strict* arc $x \xrightarrow{\ddagger} y$ represents the assertion $\sigma(x) \geq \sigma'(y)$.

We write $G \models (f, \sigma) \mapsto (g, \sigma')$ if all constraints are satisfied; we say that transition $(f, \sigma) \mapsto (g, \sigma')$ is *described by* G.

We write $x \to y \in G$ if there is an arc from x to y in G (however labeled).

An SCT instance, also known as *annotated control-flow graph* (ACG), is a CFG where every arc $c \in C_{fg}$ is annotated with a size-change graph $G_c : f \to g$.

An SCT instance is customarily denoted by $\mathcal{G}$ and often viewed as a set of SCG's, the CFG being implicitly specified.

2.2. **The SCT condition.**

Definition 2.4. A $\mathcal{G}$-*multipath* is a sequence $M = G_1 G_2 \ldots$ of elements of $\mathcal{G}$ that label a (finite or infinite) directed path in the CFG.

The path in the CFG is often denoted by cs (the abbreviation stands for call sequence—originating in a functional programming setting where transitions model calls). The multipath corresponding to cs is called M_{cs}. The *front* of the multipath is the source point of G_1, the *rear* of a finite multipath is the target of the last transition, and if both are one and the same flow-point, the multipath is referred to as *a cycle* (in fact, its underlying CFG path is a cycle).

We extend the notation $G \models s \mapsto s'$ to finite multipaths as follows:

$$G_1 G_2 \ldots G_k \models s_0 \mapsto s_k \iff (\exists s_1, \ldots, s_{k-1})(\forall i) G_i \models s_i \mapsto s_{i+1}$$

A multipath is ofen viewed as the (finite or infinite) *layered directed graph* obtained by identifying the target nodes of G_i with the source nodes of G_{i+1} (Figure 4).

Definition 2.5. Let M be a $\mathcal{G}$-multipath. A *thread* in M is a (finite or infinite) directed path in the layered directed graph representation of M. We say that the thread is *from x to y* if the thread begins with variable x and ends with y.

A thread is *complete* if it spans the length of M.

A thread is *descending* if it includes a strict arc; it is *infinitely descending* if it includes infinitely many strict arcs.

Intuitively, threads carry information along the computation, and this intuitive meaning enters the language we are using, so for example we might say that variable x is carried by

a thread to variable y (which implies that the initial value of x constrains the final value of y), or that a certain set of variables is carried by threads to some other set.

More precisely, a thread represents a sequence of values generated during a computation, that form a weakly decreasing chain in *Val*; and an infinitely descending thread indicates an infinitely-decreasing chain of values. Under the assumption of well-foundedness of *Val*, such an infinite chain cannot exist. This consideration leads to the following definition:

Definition 2.6 (The SCT Condition). $\mathcal{G}$ is said to *satisfy SCT*, or *be a positive SCT instance*, or *terminate*, if every infinite multipath contains an infinitely-descending thread.

We next formalize the manner in which the SCT condition (which is purely combinatorial) relates to a semantic notion of termination.

Definition 2.7 ($T_\mathcal{G}$). The *transition system* associated with $\mathcal{G}$ is the relation $T_\mathcal{G}$ over St defined by

$$(s, s') \in T_\mathcal{G} \iff G \models s \mapsto s' \text{ for some } G \in \mathcal{G}.$$

We say that $T_\mathcal{G}$ is *terminating* if there is no infinite chain in $T_\mathcal{G}$.

Theorem 2.8. *$T_\mathcal{G}$ is terminating if and only if $\mathcal{G}$ satisfies SCT.*

The "if" part of this theorem (soundness of the SCT criterion) follows directly from well-foundedness. For the "only if" direction see [9]. Thus, SCT is a sound and complete termination criterion for the corresponding class of transition system.

SCT is decidable (it would be a far less interesting abstraction otherwise); note that this is possible because the corresponding transition systems are restricted and only approximate real programs. A well-known way to decide SCT is the so-called *closure algorithm*, which consists of computing a transitive closure of $\mathcal{G}$ and checking idempotent graphs: we now review the pertinent definitions and facts, following [10].

Definition 2.9. The *composition* of size-change graph $G_1 : \mathtt{f} \to \mathtt{g}$ with $G_2 : \mathtt{g} \to \mathtt{h}$ is a size-change graph with source $\mathtt{f}$, target $\mathtt{h}$ and arc set $E^\downarrow \cup E^\ddagger$, where

$$
\begin{aligned}
E^\downarrow &= \{x \xrightarrow{\downarrow} z \mid x \xrightarrow{r_1} y \in G_1,\ y \xrightarrow{r_2} z \in G_2,\ r_1 \text{ or } r_2 \text{ is } \downarrow\} \\
E^\ddagger &= \{x \xrightarrow{\ddagger} z \mid x \xrightarrow{\ddagger} y \in G_1,\ y \xrightarrow{\ddagger} z \in G_2,\ x \xrightarrow{\downarrow} z \notin E^\downarrow\}.
\end{aligned}
$$

The composition is denoted by $G_1; G_2$.

Definition 2.10. Graph G is *idempotent* if $G; G = G$.

Note that an idempotent graph must have the same flow-point for both source and target, i.e., it describes a cycle in the control-flow graph.

Theorem 2.11. *Let $\mathcal{G}^+$ denote the composition-closure of $\mathcal{G}$. SCT is satisfied by $\mathcal{G}$ if and only if every idempotent graph in $\mathcal{G}^+$ has an arc $x \xrightarrow{\downarrow} x$ for some x.*

A variation in which every graph in the closure (regardless of idempotence) is tested is described in [12, 6].

2.3. **Ranking functions.**

Definition 2.12. Let $\mathcal{T}$ be a transition system over state-space St. A (global) *ranking function* for $\mathcal{T}$ is a function $\rho : St \to W$, where W is a well-founded set, such that $(s, s') \in \mathcal{T} \Rightarrow \rho(s) > \rho(s')$.

Let $P(s, s')$ be any predicate, where s, s' are free variables representing states. We write $G \models P(s, s')$ if

$$\forall s, s' \quad (G \models s \mapsto s') \Rightarrow P(s, s').$$

Definition 2.13. A (global) *ranking function* for an SCT instance $\mathcal{G}$ is a function $\rho : St \to W$, where W is a well-founded set, that constitutes a ranking function for $T_{\mathcal{G}}$. Equivalently, it satisfies $G \models \rho(s) > \rho(s')$ for every $G \in \mathcal{G}$.

For convenience, we often "Curry" ρ and write $\rho(f, \sigma)$ as $\rho_f(\sigma)$.

Complexity measures. In this paper we are interested in explicit construction of ranking functions. Thus the ranking functions will be given by expressions, combining the values of program variables with appropriate operators (such as **min**, **max** etc.). The complexity measure we are mostly interested in is the size of the expression.

Also of interest is the time to construct the expression (if the proof is constructive). Naturally, this time is lower-bounded by the expression's size.

Notations. If a flow point f has variables $x, y, \ldots$, then it is natural to write a ranking function using these variable names, e.g., $\rho_f(x, y, \ldots) = \max(x, y, \ldots)$. But technically, a ranking function is a function over St. To iron out the formality, we use notations as defined next.

Definition 2.14. Let $S \subseteq Var(f)$. Then $\mathrm{MAXVAL}_\sigma S$ is $\max\{\sigma(\mathbf{x}) \mid \mathbf{x} \in S\}$. For fixed S, this is a function of σ. Similarly, $\mathrm{MINVAL}_\sigma S$ is $\min\{\sigma(\mathbf{x}) \mid \mathbf{x} \in S\}$.

2.4. **Subclasses of SCT.** Previous work on SCT has identified certain structural subclasses of SCT as interesting. By structural, we mean that the subclass is defined by imposing a constraint on the structure of the size-change graphs. The following three subclasses will play a role in this work:

- In *Fan-in free* graphs, the in-degree of all nodes is at most 1. Fan-in free graphs are discussed in [4, 1]. A benchmark evaluation described in [4] demonstrated that such graphs occur frequently when size-change graphs are extracted from Prolog programs. In [1], it was shown that fan-in freedom makes an extended (and harder) form of SCT, called δSCT, decidable.
- In *Fan-out free* graphs, the out-degree of all nodes is at most 1. The interest in this subclass is mostly due to its being defined symmetrically to fan-in free graphs, and yet sometimes easier to work with. For example, in this paper we will handle fan-out free graphs first.
- *Strict SCT* graphs have exclusively strict arcs. This again is a class which is introduced because it is easier to work with. In particular, as shown in [9], this class admits a simple ranking-function construction; we will make use of that construction in this work.

2.5. **Multiset orderings.** In the example given in Section 1, as well as in many classical examples, it is useful to define the rank of a state as a tuple that is shown to descend lexicographically. In previous work on termination, it has been discovered that in some cases, it is useful to form not a tuple, but a multiset of values, and exhibit descent in an appropriate *multiset order*. In particular, such orders turn out useful in constructing ranking functions for SCT; this subsection presents the necessary definitions.

Definition 2.15 (Simple Multiset Order, SMO)**.** Let A, B be finite multisets over *Val*. We write $A > B$ if $|A| > |B|$ (the cardinality of A is larger) or if $|A| = |B|$ and the sets can be listed as $A = \{a_1, a_2, \dots\}$ and $B = \{b_1, b_2, \dots\}$ with $a_i \geq b_i$ for all i and $a_i > b_i$ for at least one i. We write $A \geq B$ for the non-strict variant.

For example, let $A = \{4, 3, 3, 0\}$ and $B = \{4, 3, 2, 0\}$. Then we have $A > B$. If we let $C = \{4, 3, 2, 1\}$, we have neither $A \geq C$, nor $C \geq A$: these sets are incomparable. SMO is thus a partial order.

Let $k > 0$ and suppose that $|A| = |B| = k$; it is easy to verify that $A > B$ means that a sorted listing of A is lexicographically greater than a sorted listing of B. This is true for both descending sort and ascending sort; which suggests two ways of completing SMO to a total order over k-element multisets. To complete the last example, since $\langle 4, 3, 2, 1 \rangle$ is lexicographically smaller than $\langle 4, 3, 3, 0 \rangle$, multiset C is smaller according to descending sort. However, according to ascending sort, we find that C is greater, as $\langle 1, 2, 3, 4 \rangle$ is lexicographically greater than $\langle 0, 3, 3, 4 \rangle$.

Comparing lists in descending sorted order yields the total order called *multiset order* by Dershowitz and Manna, who defined it in a more general fashion and showed its usefulness in termination proofs. Comparing ascending lists results in the so-called *dual multiset order* [4]. In this work, we will make use of both total orders. We emphasize, however, that we only use them for sets of equal cardinality (which simplifies their definitions). If $|A| > |B|$, we shall always consider A to be bigger.

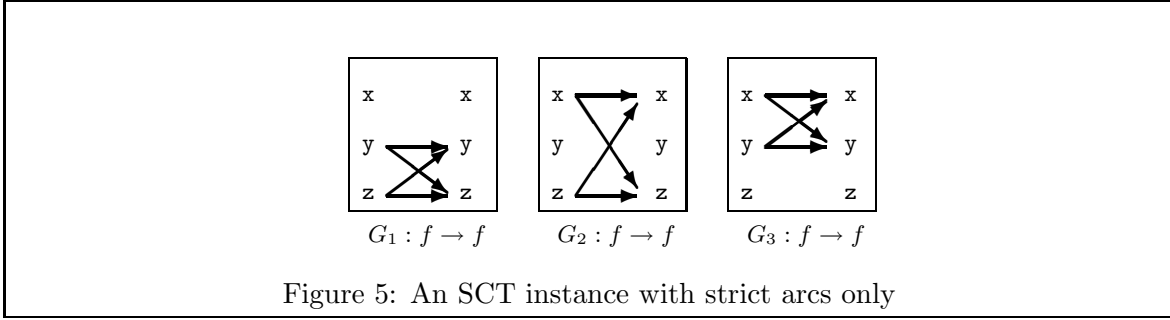A total order is needed, in particular, for the definition of min and max operators.

Definition 2.16. Let A, B be finite multisets over *Val*. We define $\min(A, B)$ and $\max(A, B)$ as follows: if $|A| \neq |B|$, then $\min(A, B)$ is the smaller multiset and $\max(A, B)$ is the larger. If $|A| = |B|$, $\min(A, B)$ is the smaller under dual multiset order, while $\max(A, B)$ is the bigger under multiset order.

The definitions extend naturally to define minimum and maximum over a finite set of multisets. In all cases, the operator can be implemented by choosing the lexicographic minimum, or maximum, among the tuples that represent the multisets (with elements in ascending order for **min**, descending order for **max**). The choice of two different orderings to define **min** and **max** may seem strange, but it will be seen to work best later in the paper.

3. Some Previous Results

This section summarizes some previous work on SCT, including definitions and constructions that we shall use.

$$G_1 : f \to f \qquad G_2 : f \to f \qquad G_3 : f \to f$$

Figure 5: An SCT instance with strict arcs only

3.1. **Thread preservers.** The concept of thread preservers was introduced in [4]. We cite the definition and a useful theorem.

Let $\mathcal{V} = \bigcup_f Var(f)$, the combined set of variables in the whole abstract program.

Definition 3.1. Let $\mathcal{G}$ be an ACG and $\mathcal{V}$ its set of variables. A set $P \subseteq \mathcal{V}$ is called a *thread preserver* for $\mathcal{G}$ if for every $G \in \mathcal{G}$ where $G : f \to g$, it holds that whenever $x \in (Var(f) \cap P)$, there is $x \to y \in G$ for some $y \in P$.

It is easy to see that the set of thread preservers of a given ACG is closed under union. Hence, there is always a unique maximal thread preserver (MTP) for $\mathcal{G}$, which we denote by $\mathrm{MTP}(\mathcal{G})$. It is further shown in [4] that given a standard representation of $\mathcal{G}$, $\mathrm{MTP}(\mathcal{G})$ can be found in linear time. This is significant because the MTP is useful—among else for constructing ranking functions.

3.2. **Ranking functions for strict SCT.** Throughout this subsection, $\mathcal{G}$ is presumed strongly connected.

The following theorem is from [9]:

Theorem 3.2. *Let $\mathcal{G}$ be a terminating, strict SCT instance. There exists an indexed set $\{S_f\}$, where for every $f \in F$, S_f is a set of subsets of $Var(f)$, such that the function*

$$\rho(f, \sigma) = \min_{X \in S_f} \mathrm{MAXVAL}_\sigma X$$

is a ranking function for $\mathcal{G}$.

For example, consider the SCT instance in Figure 5. It has the ranking function

$$\rho(f, \sigma) = \min(\mathrm{MAXVAL}_\sigma\{x, y\}, \mathrm{MAXVAL}_\sigma\{x, z\}, \mathrm{MAXVAL}_\sigma\{y, z\}).$$

The correctness of this function can be verified, as usual, by checking graph by graph and assuming in turn each ordering of the values of the variables.

The worst-case size of a function of the above form is exponential, related to the number of different subsets of $Var(f)$, which is 2^n. For more details, see [9, § 3.1]. Next, we give two special cases of particular interest, which are already implicit in [4]. The first case is that of *fan-out free* graphs:

Theorem 3.3. *If $\mathcal{G}$ is a terminating, strict, fan-out free SCT instance, then $\mathcal{G}$ has a non-empty thread-preserver; and for any such preserver P, the function*

$$\rho(f, \sigma) = \mathrm{MINVAL}_\sigma(Var(f) \cap P)$$

is a ranking function for $\mathcal{G}$.

The theorem follows from the following two lemmas. The first we only cite, referring the reader to [4, § 6] for a proof. The second one we prove, because the proof is simple and clarifies the significance of thread preservers in connection with ranking.

Lemma 3.4. *Suppose that $\mathcal{G}$ is strict and fan-out free. Then $\mathcal{G}$ is size-change terminating if and only if $\mathcal{G}$ has a non-empty thread-preserver.*

Lemma 3.5. *Suppose that $\mathcal{G}$ has a non-empty thread preserver P, and let*

$$\rho(f, \sigma) = \mathrm{MINVAL}_\sigma(\mathit{Var}(f) \cap P).$$

For all $G \in \mathcal{G}$, $G \models \rho(s) \geq \rho(s')$. If $\mathcal{G}$ is strict, then $G \models \rho(s) > \rho(s')$.

Proof. Let $G \models (f, \sigma) \mapsto (f', \sigma')$. By the definition of a thread preserver, we have, for all $x \in (\mathit{Var}(f) \cap P)$, $\sigma(x) \geq \sigma'(y)$ for some $y \in \mathit{Var}(f') \cap P$. If x is such that $\sigma(x)$ is minimum (i.e., $\mathrm{MINVAL}_\sigma(\mathit{Var}(f) \cap P)$ is $\sigma(x)$), we have $\rho(f, \sigma) = \sigma(x)$, whereas $\rho(f', \sigma') \leq \sigma(y)$. Combining the three relations, we get $\rho(f', \sigma') \leq \rho(f, \sigma)$.

If $\mathcal{G}$ is strict, then $\sigma(x) > \sigma'(y)$ and the inequality becomes strict. □

The case of *fan-in free graphs* is symmetric to that of fan-out free graphs. In order to exploit this symmetry, we use the technique of *transposing* the size-change graphs.

Definition 3.6. If $G : f \to g$ is a size-change graph, G^t denotes its transposition, which is a size-change graph with source g, target f, and arcs $\{y \to x \mid x \to y \in G\}$. For a set $\mathcal{G}$ of size-change graphs, $\mathcal{G}^t = \{G^t \mid G \in \mathcal{G}\}$.

Observation 3.7 ([4])**.** $\mathcal{G}$ satisfies SCT if and only if $\mathcal{G}^t$ does.

Clearly, G is fan-in free if and only if G^t is fan-out free. Now we can use this for a ranking-function construction.

Theorem 3.8. *If $\mathcal{G}$ is a terminating, strict, fan-in free SCT instance, then $\mathcal{G}^t$ has a non-empty thread-preserver; and for any such preserver P, the function*

$$\rho(f, \sigma) = \mathrm{MAXVAL}_\sigma(\mathit{Var}(f) \cap P)$$

is a ranking function for $\mathcal{G}$.

The fact that $\mathcal{G}^t$ has a non-empty thread-preserver follows from Observation 3.7 and Lemma 3.4. The correctness of the ranking function then follows from the next lemma.

Lemma 3.9. *Suppose that $\mathcal{G}^t$ has a non-empty thread preserver P. And let*

$$\rho(f, \sigma) = \mathrm{MAXVAL}_\sigma(\mathit{Var}(f) \cap P).$$

For all $G \in \mathcal{G}$, $G \models \rho(s) \geq \rho(s')$. If $\mathcal{G}$ is strict, then $G \models \rho(s) > \rho(s')$.

Proof. Let $G \models (f, \sigma) \mapsto (f', \sigma')$. By the definition of a thread preserver (but noting that it is a G^t thread preserver!), we have, for all $y \in (\mathit{Var}(f') \cap P)$, $\sigma(x) \geq \sigma'(y)$ for some $x \in \mathit{Var}(f) \cap P$. For y such that $\sigma(y)$ is maximum, we have $\rho(f', \sigma') = \sigma(y)$, whereas $\rho(f, \sigma) \geq \sigma(x)$. Combining the three relations, we get $\rho(f, \sigma) \geq \rho(f', \sigma')$.

If $\mathcal{G}$ is strict, then $\sigma(x) > \sigma'(y)$ and the inequality becomes strict. □

4. Fan-out Free SCT

We are given a fan-out free positive SCT instance $\mathcal{G}$. We assume that $\mathcal{G}$ is strongly connected and that $|Var(f)| = n$ for all $f \in F$, and let $m = |F|$. We shall construct a ranking function for $\mathcal{G}$, and then discuss its size (getting an upper bound which is later shown to be tight). A brief outline of the construction follows:

- Transform $\mathcal{G}$ into a strict, fan-out free SCT instance.
- Use Theorem 3.3.
- Optionally, optimize the ranking function for size (without this stage, the desired upper bound on size may fail to hold).

Next, the construction is described in detail, along with proofs and some illustrations. A complete demonstration of the process for a (very small) example can be found at the end of the section, along with some comments regarding the implementation of the algorithm (which is initially described in a very abstract manner, just to make the theorems clear).

4.1. **The basic construction.**

Definition 4.1 (Vectors). For flow-point $f \in F$ and positive integer B, V_f^B is the set of tuples $\mathbf{v}_f = \langle v_1, v_2, \dots \rangle$ of even length, where every odd position is a non-empty subset of $Var(f)$, together constituting a partition of the latter; and every even position is an integer between 0 and B.

Remark. The distinction between a vector and its components is indicated here by font. Thus, v_i is the ith entry of $\mathbf{v}$, while $\mathbf{v}_i$ is the ith vector in some sequence of vectors. For notational convenience, we may make use of a double-meaning expression $|v_i| > |u_i|$, which means, if i is odd, that the set v_i contains more elements than u_i; and if i is even, that the integer v_i is greater.

Definition 4.2. The value of $\mathbf{v} \in V_f^B$ in a given program state (f, σ), denoted $\sigma(\mathbf{v})$, is obtained by substituting the values of variables according to σ, so every subset of variables becomes a multiset of data values. This results in a tuple with multisets and integers in odd and even positions, respectively. Such tuples are compared lexicographically, where multisets are compared according to one of the multiset orders (we use SMO, as long as a total order is not required). We define the **min** and **max** operators on vectors by the lexicographic extension of the corresponding total multiset order, according to Definition 2.16.

In the rest of this section, $B = m \cdot 2^n$. Since B is fixed, V_f^B may be written as V_f.

Definition 4.3. For $S \subseteq Var(f)$ and a size-change graph $G : f \to g$, define $\text{Im}(S, G)$ to be the set of y such that $x \to y \in G$ with $x \in S$.

Observe that, as we are only dealing with fan-out free graphs in this construction, $|\text{Im}(S, G)| \leq |S|$, with equality only if we have a *one-to-one* correspondence, where for every element of S (respectively $\text{Im}(S, G)$) there is a single arc in G connecting it to the other set. It is quite easy to see, that such correspondence implies a (weak) SMO descent from S to $\text{Im}(S, G)$ (if at least one of the connecting arcs is strict, we have strict descent).

Definition 4.4. Let $\mathbf{v} \in V_f$, and $G : f \to g$. For an odd position i in $\mathbf{v}$, let

$$\text{Im}_i(\mathbf{v}, G) = \text{Im}(v_i, G) \setminus \bigcup_{\text{odd } j < i} \text{Im}(v_j, G).$$

We say that position i is *descending* if $|\mathrm{Im}_i(\mathbf{v}, G)| < |v_i|$ or $|\mathrm{Im}_i(\mathbf{v}, G)| = |v_i|$ and there is a strict arc in G from $x \in v_i$ to $y \in \mathrm{Im}_i(\mathbf{v}, G)$.

Note that the only other possibility is $|\mathrm{Im}_i(\mathbf{v}, G)| = |v_i|$ and all arcs from v_i to $\mathrm{Im}_i(\mathbf{v}, G)$ non-strict. So, a descending position's value either strictly descends in multiset order, or at least weakly descends. In the next definition, we use the numeric positions in order to make the vector decrease strictly when there is no descent in the set-valued positions. We also use them to avoid having empty sets in the set-valued positions.

Definition 4.5. Given a size-change graph $G : f \to g$, and $\mathbf{v} \in V_f$, $\mathrm{next}(\mathbf{v}, G) \in V_g$ is defined by cases, as follows.

Case N1: If i is the first descending position and $|\mathrm{Im}_i(\mathbf{v}, G)| > 0$,

$$\mathrm{next}(\mathbf{v}, G) = \langle \mathrm{Im}_1(\mathbf{v}, G), v_2, \mathrm{Im}_3(\mathbf{v}, G), \dots, \mathrm{Im}_i(\mathbf{v}, G), B, S, B \rangle$$

where S is the set of g variables not occurring up to position i; if S turns out to be empty, omit the final suffix S, B.

Case N2: If i is the first descending position and $|\mathrm{Im}_i(\mathbf{v}, G)| = 0$,

$$\mathrm{next}(\mathbf{v}, G) = \langle \mathrm{Im}_1(\mathbf{v}, G), v_2, \mathrm{Im}_3(\mathbf{v}, G), \dots, \mathrm{Im}_{i-2}(\mathbf{v}, G), v_{i-1} - 1, S, B \rangle$$

where S is the set of g variables not occurring up to position $i - 2$; note that it cannot be empty. The above expression assumes that $i > 1$ and $v_{i-1} > 0$. If either of these conditions is not met, $\mathrm{next}(\mathbf{v}, G)$ is *undefined*.

Case N3: No position is descending. The last position is v_i (an integer).

$$\mathrm{next}(\mathbf{v}, G) = \langle \mathrm{Im}_1(\mathbf{v}, G), v_2, \mathrm{Im}_3(\mathbf{v}, G), \dots, \mathrm{Im}_{i-1}(\mathbf{v}, G), v_i - 1 \rangle$$

assuming that $v_i > 0$; if $v_i = 0$, $\mathrm{next}(\mathbf{v}, G)$ is *undefined*.

Note that a size-change arc of G that leaves a variable $x \in v_i$ never reaches a variable in a higher position of $\mathrm{next}(\mathbf{v}, G)$; it may reach position i or a lower one. Here is a more substantial observation:

Observation 4.6. In a program transition $(f, \sigma) \overset{c}{\to} (g, \sigma')$, the value of $\mathrm{next}(\mathbf{v}, G_c)$ in (g, σ') (if defined) is strictly smaller than the value of $\mathbf{v}$ in (f, σ).

Thus, we can use these vectors to construct an instance of strict SCT. Note that the descent is justified by the definition of next and by size-change graph G_c. Therefore, a ranking function built on the base of this descent can be statically verified to decrease, based on $\mathcal{G}$.

Definition 4.7. $\mathcal{R}$ is an all-strict, fan-out free SCT instance with CFG as in $\mathcal{G}$, where the variables for flow-point f are the vectors V_f, and the size-change graph G_c^R for arc $c \in C_{fg}$ has arcs $\mathbf{v} \to \mathrm{next}(\mathbf{v}, G_c)$ for all $\mathbf{v} \in V_f$ such that $\mathrm{next}(\mathbf{v}, G_c)$ exists.

Clearly, $\mathcal{R}$ is fan-out free. Note that this implies that, given a multipath and a specific vector at its source, a unique thread can be followed from that vector until it either stops at the end of the multipath or reaches a vector with no outcoming arc. Such a thread is a chain of vectors obtained by repeated application of next. For an example, Figure 6 shows three size-change graphs of an instance $\mathcal{G}$ and Figure 7 shows a single thread from an $\mathcal{R}$-multipath. Note that there is lexicographic descent at each step.
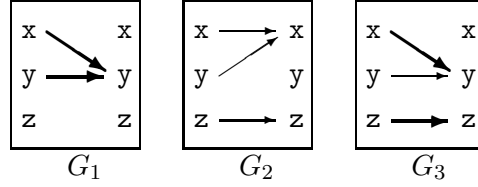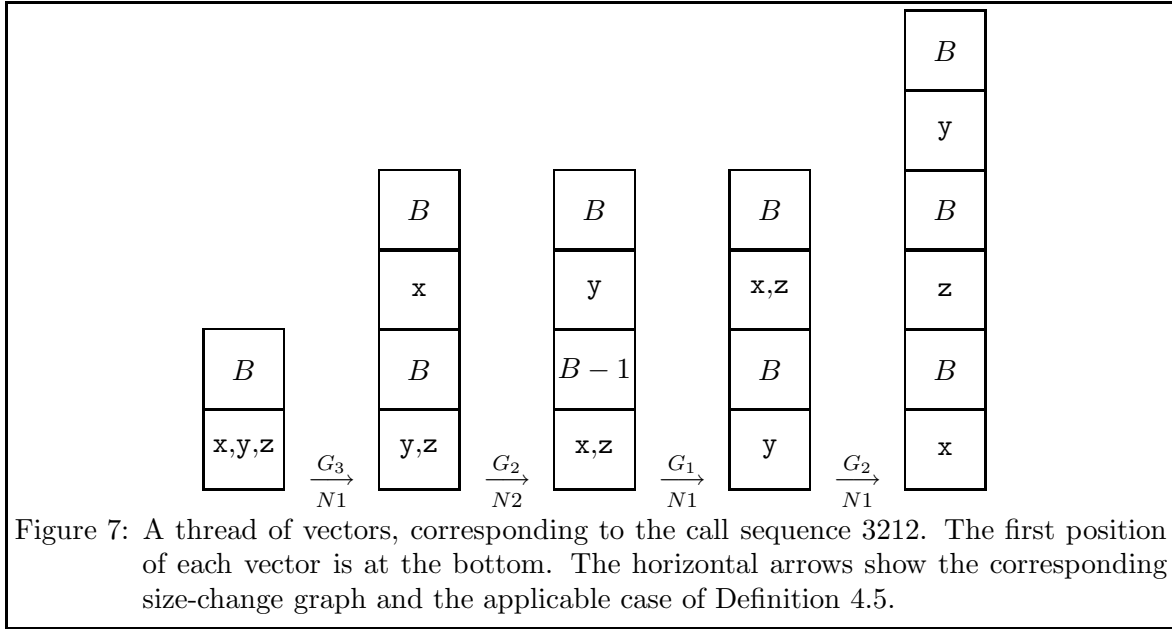
Figure 6: A fan-out free SCT instance.

Figure 7: A thread of vectors, corresponding to the call sequence 3212. The first position
of each vector is at the bottom. The horizontal arrows show the corresponding
size-change graph and the applicable case of Definition 4.5.

We now introduce some notation and concepts used later in analysing $\mathcal{R}$. Let $M^G = G_1 G_2 \ldots$ denote a $\mathcal{G}$-multipath, and M^R the corresponding $\mathcal{R}$-multipath (following the same CFG path). Let τ denote a thread in M^R, namely a sequence $\mathbf{v}_0, \mathbf{v}_1, \ldots$ such that $\mathbf{v}_j = \text{next}(\mathbf{v}_{j-1}, G_j)$. The passage from $\mathbf{v}_{j-1}$ to $\mathbf{v}_j$ will be called *a step* and written compactly as $\mathbf{v}_{j-1} \overset{G_j}{\to} \mathbf{v}_j$.

Definition 4.8. The i-band of τ is the sequence resulting from trimming all vectors in τ to their i-prefix, i.e., the first i positions. We say that τ is i-stable if for all steps $\mathbf{v} \overset{G}{\to} \mathbf{v}'$ in τ, no odd position $k \leq i$ is descending.

We denote by $P_i(\mathbf{v})$ the set of variables appearing in the i-prefix of $\mathbf{v}$.

Lemma 4.9. *Suppose that i is even, and τ is an $\mathcal{R}$-thread with a stable i-band. Then none of the numeric (even) positions below i changes along τ, while the last one (position i) may decrease, but cannot increase.*

Proof. Straight-forward from definition of the next relation and i-stability. □

On the other hand, if the i-band is not stable in τ, then either position i or a lower numeric position must be reset to B once or more along τ.
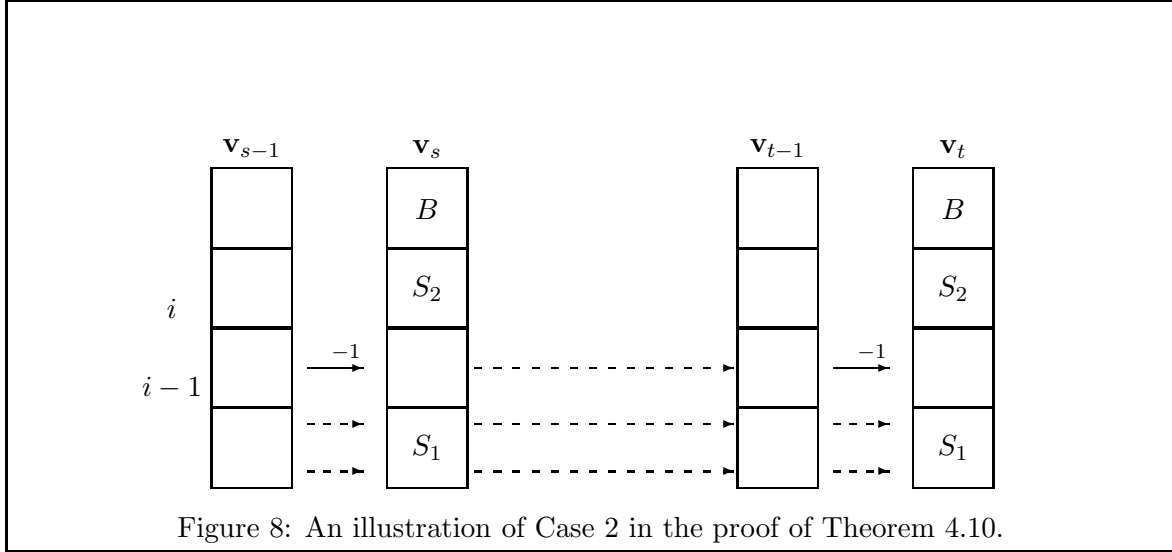
Theorem 4.10. $\mathcal{R}$ *satisfies SCT.*

Figure 8: An illustration of Case 2 in the proof of Theorem 4.10.

Proof. Let f_0 be the initial function of $\mathcal{G}$ and let $\mathbf{v}_0 = \langle Var(f_0), B \rangle$. We claim that in every $\mathcal{R}$ multipath (in particular, an infinite one) starting at f_0 there is a complete thread starting at $\mathbf{v}_0$. Since $\mathcal{R}$ is strict, this means that every infinite multipath has an infinitely-descending thread, hence SCT is satisfied.

Suppose to the contrary that there is a counter-example M^R. In fact, let M^R be a shortest counter-example. Then M^R ends in a graph G_c^R and there is a thread τ leading from $\mathbf{v}_0$ at the front of M^R up to a node $\mathbf{v}$ on the source side of G_c^R such that $\mathrm{next}(\mathbf{v}, G_c)$ is undefined.

Let us review the cases in which $\mathrm{next}(\mathbf{v}, G_c)$ is undefined:

(1) $|\mathrm{Im}_1(\mathbf{v}, G)| = 0$.
(2) The first descending position is i and $|\mathrm{Im}_i(\mathbf{v}, G)| = 0$ and $v_{i-1} = 0$.
(3) No position is descending and the vector ends with a 0.

We next undertake to derive a contradiction in each of these cases.

Case 1. This case implies that there is no complete thread in M^G; indeed, when looking at the vectors of τ, we see that initially all the variables are in the lowest position, and all threads must stay there (review the definition of $|\mathrm{Im}_i(\mathbf{v}, G)|$); by the definition of this case, none of them survives to the end of the multipath.

Since $\mathcal{G}$ is strongly connected, it is possible to complete M^G by adding size-change graphs, if necessary, so that the multipath ends at f_0. We thus obtain a cycle without any complete thread, which contradicts the SCT property of $\mathcal{G}$ (repeat the cycle forever and obtain an infinite multipath without infinite descent).

Case 2. $|\mathrm{Im}_i(\mathbf{v}, G_c)| = 0$, $v_{i-1} = 0$ while for all odd $j < i$, $|\mathrm{Im}_j(\mathbf{v}, G_c)| = |v_j|$ and there are only non-strict arcs between v_j and $\mathrm{Im}_j(\mathbf{v}, G_c)$.

Observe that v_{i-1} can only reach 0 by being decreased repeatedly, from the last point where it had a value of B down to zero; this means that the i-band of τ is stable.

Informally speaking, we can ignore the i-band and look at the vectors starting at the ith position, and there we will find a situation similar to Case 1. For this reasoning to be correct, it is necessary that the set of variables within the i-band be the same at the

beginning and the end of the "bad" multipath. We will use a pigeon-hole argument to show that such a situation must occur.

Consider the steps $\mathbf{v}_{t-1} \overset{G_t}{\to} \mathbf{v}_t$ of τ where position $i-1$ is decremented. Label each such step with its target flow-point, say f, and the subset $S_1 = P_{i-1}(\mathbf{v}_t) \subseteq Var(f)$. The value $B = m \cdot 2^n$ guarantees that some pair (f, S_1) must occur twice. Choose two positions $s < t$ that get the same label (f, S_1). Then $\text{next}(\mathbf{v}_{s-1}, G_s)$ as well as $\text{next}(\mathbf{v}_{t-1}, G_t)$ are defined by (N2). Therefore, $\mathbf{v}_s$ has $S_2 = Var(f) \setminus S_1$ in position i. In addition, $|\text{Im}_i(\mathbf{v}_{t-1}, G_t)| = 0$, that is, the set in position i has no outgoing arcs in G_t.

Now, consider the $\mathcal{G}$-multipath $C = G_{s+1} \ldots G_t$, starting and ending at f (Figure 8). The set S_1 is carried by non-strict threads of C unto itself, while no complete thread starts and ends within S_2. Conclusion: repeating this cycle infinitely many times, there will be no infinite descent, contradicting the termination of $\mathcal{G}$.

Case 3. No position in $\mathbf{v}$ is descending and the vector ends with a 0.

This case can be seen as a sub-case of Case 2, where S_1 includes all variables. $\square$

Theorem 4.11. *Suppose that $\mathcal{G}$ is a fan-out free, positive SCT instance. Let $B = m \cdot 2^n$. There exists an indexed set $\{S_f\}$, where $S_f \subseteq V_f^B$ for every $f \in F$, such that the function $\rho_f(\sigma) = \text{MINVAL}_\sigma S_f$ is a ranking function for $\mathcal{G}$. This ranking function can be effectively constructed given $\mathcal{G}$.*

Proof. Since $\mathcal{R}$ is all-strict, fan-out free, and satisfies SCT, Theorem 3.3 applies. By constructing $\mathcal{R}$ and computing its maximal thread preserver, we find the sets of vectors whose minima comprise the ranking function. $\square$

Practically, $\mathcal{R}$ need not be constructed to its full doubly-exponential size; see Section 4.5.

4.2. **A useful observation.** We note a fact that will prove useful in the sequel. Suppose that we modify the SMO into a partial order SMO† by defining sets of different cardinalities to be incomparable. We can still carry out our construction. The trick is simple: if we replace a set S by the pair $\langle |S|, S \rangle$, the intended order is restored. An extra numeric position before a set-valued position v_i is not actually needed, because it can be merged into the numeric position v_{i-1} (replacing $\langle v_{i-1}, |v_i| \rangle$ with $v_{i-1}n + |v_i|$; the upper bound B changes into Bn). The case of v_1 is different: here the prefix $|v_1|$ is not necessary at all, because in an infinite thread, the size $|v_1|$ must be eventually constant.

4.3. **The size of the ranking function.** If the size of the ranking function (more precisely, the expression for that function) is of interest, the expression that results of the previous construction should be optimized by eliminating redundancies, as explained next.

Definition 4.12. Let $\mathbf{v}, \mathbf{u} \in V_f$. The relation $\mathbf{v} \prec \mathbf{u}$ holds if there is an index i, such that for all $j < i$, $v_j = u_j$, while $|v_i| < |u_i|$.

Observation 4.13. If $\mathbf{v} \prec \mathbf{u}$ then, regardless of the values of variables, the value of $\mathbf{v}$ is lexicographically smaller than the value of $\mathbf{u}$. We say therefore that $\mathbf{v}$ *dominates* $\mathbf{u}$.

Clearly, if there are dominated vectors in the ranking expression, they can be dropped. A set without dominated vectors is said to be *in normal form*.

Lemma 4.14. *Any set $R \subseteq V_f$ in normal form satisfies $|R| \leq n!$.*

Proof. Observe that all first positions of vectors $\mathbf{v} \in R$ must contain sets of the same size k_1. We prove the lemma by induction on $n - k_1$. If $k_1 = n$ then there is just one set and clearly $|R| = 1$. If $k_1 < n$ there are at most $\binom{n}{k_1}$ different sets in the first position. Given any choice S_1 for the first position, the number in the second position is unique (by normality). If we choose from R just the vectors beginning with S_1 and drop the first two positions, we obtain a normal set over $n - k_1$ variables. By the induction hypothesis, its size is bounded by $(n - k_1)!$ and the bound on $|R|$ readily follows. □

Corollary 4.15. *Every fan-out free, terminating SCT instance has a ranking function of form $\rho_f(\sigma) = \mathrm{MINVAL}_\sigma S_f$ with $S_f \subseteq V_f$ in normal form. In particular, $|S_f| \leq n!$.*

We can also simplify the form of the function—expressing it without the use of multisets. Let $\widetilde{V}_f^B$ be the set of V_f^B vectors in which all odd positions are singletons. When working exclusively with such vectors, the multiset concept is redundant. One can define the value of a position holding $\{x\}$ as $\sigma(x)$ and consider the vectors to have values in $(Val \times \mathbb{N})^n$.

For $\mathbf{v} \in V_f$, a *simplification* of $\mathbf{v}$ is obtained as follows. For every odd position i, replace the set v_i with a list of its elements (if $|v_i| > 1$, there are different ways to do that, so there are many simplifications). Insert 0 between any pair of consecutive non-numerical positions, to obtain an element of $\widetilde{V}_f^B$. For $R \subseteq V_f$, let $\widetilde{R}$ be the set of all simplifications of $\mathbf{v} \in R$.

Lemma 4.16. *Suppose that $S_f \subset V_f^B$ and $S_g \subset V_g^B$ are normal. If $\mathrm{MINVAL}_\sigma S_f > \mathrm{MINVAL}_{\sigma'} S_g$, then also $\mathrm{MINVAL}_\sigma \widetilde{S}_f > \mathrm{MINVAL}_{\sigma'} \widetilde{S}_g$.*

Proof. Let $\widetilde{\mathbf{v}} \in \widetilde{S}_f$ be the vector of minimal value (under σ). Consider $\mathbf{v}$; the assumption $\mathrm{MINVAL}_\sigma S_f > \mathrm{MINVAL}_{\sigma'} S_g$ implies that there is a vector $\mathbf{u} \in S_g$ such that $\sigma'(\mathbf{u}) < \sigma(\mathbf{v})$. Note that this is a lexicographic comparison. Thus there is an index i such that for all $j < i$, $\sigma'(\mathbf{u})_j = \sigma(\mathbf{v})_j$, while $\sigma'(\mathbf{u})_i < \sigma(\mathbf{v})_i$.

The equalities at positions $j < i$ mean that for all the sets in such positions, the elements of u_j can be arranged to match elements of v_j of equal value as they appear in $\widetilde{\mathbf{v}}$. Thus we build a simplification $\widetilde{\mathbf{u}}$ whose value is identical to that of $\widetilde{\mathbf{v}}$ up to position i. If this position is numeric, then we can clearly complete the simplification and obtain $\sigma(\widetilde{\mathbf{v}}) > \sigma'(\widetilde{\mathbf{u}})$. If position i is a set, the multiset inequality $\sigma'(\mathbf{u})_i < \sigma(\mathbf{v})_i$ means that, if the elements of both vectors are arranged in ascending order of value, the vector obtained from u_i will be lexicographically smaller. Observe that among all listings of the elements of a multiset, the ascending list *is* lexicographically smallest. Therefore, the elements of v_i must appear in $\widetilde{\mathbf{v}}$ in exactly that order (as it is minimum in $\widetilde{S}_f$). Now, arranging the elements of u_i in corresponding order, we obtain a simplification $\widetilde{u}$ that is smaller than $\widetilde{v}$.

Thus some element of $\widetilde{S}_g$ has smaller value than $\widetilde{v}$, which shows that $\mathrm{MINVAL}_{\sigma'} \widetilde{S}_g$ is definitely smaller than $\widetilde{v}$. □

Theorem 4.17. *Every fan-out free, positive SCT instance $\mathcal{G}$ has a ranking function of form $\rho_f(\sigma) = \mathrm{MINVAL}_\sigma S_f$ with $S_f \subseteq \widetilde{V}_f^B$ and $|S_f| \leq n!$.*

Proof. We start with a ranking function as claimed in Theorem 4.11 and replace every set S_f by its simplification; then we remove dominated vectors to obtain a normal set. □

4.4. **When the CFG is not strongly connected.** For completeness, let us explain how ranking functions of the kind constructed in this paper should be adjusted when the control-flow graph is not strongly connected.

Suppose that the CFG of $\mathcal{G}$ consists of several strongly connected components (SCCs). Let $C_1, \dots, C_k$ be a reverse topological ordering of the components. If the size-change graphs in every component are fan-out (or fan-in) free, our constructions yields a function $\rho_f(\sigma)$ that decreases in every transition within a component (note that we construct the function separately for each component, but it can be considered as one function; no conflicts arise because every flow-point f belongs to a unique component).

Next, define $\rho'(s)$ for any state $s = (f, \sigma)$, where $f \in C_i$, by prepending i to $\rho_f(\sigma)$ (this gives a vector with an extra numeric position at its beginning). It is easy to see that ρ' is a ranking function for $\mathcal{G}$.
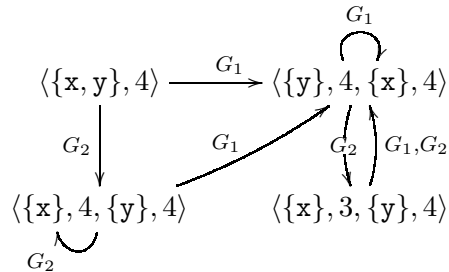
4.5. **An example, and a simplification of the algorithm.** Next, we carry out the construction for a very small example. Small, but hopefully illustrative. In fact, the example will illustrate how the construction can be made simpler and much more efficient than a literal implementation of the proof of Theorem 4.11. After explaining the example, we will also formulate the simplification in general terms.

Our example is an instance with $F = \{f\}$, $Var(f) = \{\mathtt{x}, \mathtt{y}\}$ and $\mathcal{G} = \{G_1, G_2\}$, where

$$G_1 = \{\mathtt{x} \xrightarrow{\downarrow} \mathtt{y}, \ \mathtt{y} \xrightarrow{\downarrow} \mathtt{y}\}, \ G_2 = \{\mathtt{x} \xrightarrow{\downarrow} \mathtt{x}, \ \mathtt{y} \xrightarrow{\overline{\downarrow}} \mathtt{x}\}.$$

We have $B = m \cdot 2^n = 4$. We will not construct $\mathcal{R}$ in full, which would have meant (if we followed Definition 4.7 literally) creating an instance with $|V_f| = 36$ variables. Instead, we note that according to the proof of Theorem 4.10, it suffices to find the vectors reachable from $\mathbf{v}_0 = \langle Var(f_0), B \rangle$ by applications of next; this will be a smaller sub-instance of $\mathcal{R}$ which satisfies the theorem.

So, the actual procedure is as follows: we initialize a collection S of vectors to $\{\mathbf{v}_0\}$. We repeatedly compute, for each vector $\mathbf{v} \in S$, and $G_i \in \mathcal{G}$, the result of $\mathrm{next}(\mathbf{v}, G_i)$, adding it to S, until S does not grow any further. We describe the result of this process as a graph, where the arcs out of $\mathbf{v}$ describe the applications of $\mathrm{next}(\mathbf{v}, G_i)$ in the construction:

The set of all four vectors is a thread preserver, as each of them has an outgoing arc both under G_1 and under G_2. We conclude that

$$\rho(\mathtt{x}, \mathtt{y}) = \min(\langle\{\mathtt{x}, \mathtt{y}\}, 4\rangle, \ \langle\{\mathtt{y}\}, 4, \{\mathtt{x}\}, 4\rangle, \ \langle\{\mathtt{x}\}, 4, \{\mathtt{y}\}, 4\rangle, \ \langle\{\mathtt{x}\}, 3, \{\mathtt{y}\}, 4\rangle)$$

is a ranking function. Removing dominated vectors, we reduce the expression to

$$\rho(\mathtt{x}, \mathtt{y}) = \min(\langle\{\mathtt{y}\}, 4, \{\mathtt{x}\}, 4\rangle, \ \langle\{\mathtt{x}\}, 3, \{\mathtt{y}\}, 4\rangle).$$

The fact that the set of vectors constructed, S, constituted a thread preserver, is not an incident. We next demonstrate this in general. This means that the way of computing S not only economizes on the size of $\mathcal{R}$, it also obviates the need for an MTP computation.

Theorem 4.18. *Let $S \subseteq V_f$ be the closure of the set $\{\mathbf{v}_0\}$, where $\mathbf{v}_0 = \langle Var(f_0), B \rangle$, under the operators $\mathrm{next}(\mathbf{v}, G)$ for all $G \in \mathcal{G}$. This set is a thread preserver in $\mathcal{R}$.*

Proof. By the definition of S, if $\mathbf{v} \in V_f$ is in S, this is because there exists a sequence $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_k = \mathbf{v}$ such that for all i, $\mathbf{v}_i = \mathrm{next}(\mathbf{v}_{i-1}, G_i)$ for an appropriate G_i. Thus, in the $\mathcal{R}$-multipath $M = G_1 G_2 \ldots G_k$, vector $\mathbf{v}$ lies on a thread τ emanating from $\mathbf{v}_0$. The crucial observation now is that, *since $\mathcal{R}$ is fan-out free, there is just one such thread.* This fact was not used in the proof of Theorem 4.10, but it means that we can deduce from the proof that this particular thread τ can be continued in any $\mathcal{R}$-multipath extending M, say MG_{k+1}. Thus, for any applicable G_{k+1}, $\mathrm{next}(\mathbf{v}, G_{k+1})$ is defined—and has to be in S.

We conclude that S has the property of a thread preserver. □

This technique can be pushed a little further, as shows the next theorem.

Theorem 4.19. *Considering S as the node set of a directed graph (S, A) (with arcs $\mathbf{v} \to \mathrm{next}(\mathbf{v}, G)$), let C be a sink SCC of this graph. Then C is a thread preserver.*

Proof. The thread-preservation follows directly from the definition of S plus the fact that no arc leaves a sink SCC. □

Thus, a standard SCC algorithm (which runs in linear time [7]) suffices for obtaining a thread preserver, which can be smaller than the MTP, since it is actually a *minimal* thread preserver (as the interested reader may verify). In the last example, this TP consists of the two vectors that wound up in the final ranking function. It is not true in general, however, that the SCC will be a normal set (free of dominated vectors), as was the case with this example. A counter-example is given by $\mathcal{G} = \{G_1, G_2\}$, where

$$G_1 = \{\mathtt{x} \xrightarrow{\downarrow} \mathtt{x},\ \mathtt{y} \xrightarrow{\ddagger} \mathtt{y},\ \mathtt{z} \xrightarrow{\downarrow} \mathtt{y}\},\ G_2 = \{\mathtt{y} \xrightarrow{\downarrow} \mathtt{z},\ \mathtt{z} \xrightarrow{\downarrow} \mathtt{z}\}.$$

An additional implementation tip. The value of B used in Theorem 4.11, $m \cdot 2^n$, becomes unwieldy if n is large. In fact, it is an overestimate. Even theoretically, the range 0 to B will never be fully used (the interested reader is invited to prove it), and in most cases a much smaller range will be needed. The right way to implement the numeric positions is by inverting the interval $[0, B]$, so that they are initially 0 and increasing, instead of starting at B and decreasing. For preserving the natural order on numbers, invert again once the true range necessary for an instance has thus been discovered.

Other optimizations. The reader may have noticed that the function:

$$\rho(\mathtt{x}, \mathtt{y}) = \min(\langle \{\mathtt{y}\}, 4, \{\mathtt{x}\}, 4 \rangle,\ \langle \{\mathtt{x}\}, 3, \{\mathtt{y}\}, 4 \rangle)$$

can be simplified to

$$\rho(\mathtt{x}, \mathtt{y}) = \min(\langle \{\mathtt{y}\}, 4 \rangle,\ \langle \{\mathtt{x}\}, 3 \rangle)$$

while remaining a ranking function. We have not generalized this observation or formulated a procedure to find such savings.

5. Fan-in Free SCT

The fan-in free class of SCT instances is symmetric in nature to the fan-out free class. This symmetry has been used in [4] where it was also observed that transposing graphs, which clearly makes fan-in into fan-out (and vice versa), also has to do with exchanging **min**-descent with **max**-descent; this is illustrated in the simple results cited in Section 3.2.

In this section, we show how to use transposition for applying the construction and results of Section 4 to fan-in free graphs. The arguments here are somewhat subtler than those of Section 3.2, since they involve the semantic connection between $\mathcal{G}$ and $\mathcal{R}$. To do this precisely, we have to review some of our concepts and results with an eye to more generality.

We begin by noting that the SCT condition (Section 2.2) is completely independent of the semantic interpretation of size-change graphs, involving the order relation on Val, to which we now assign the notation $\leq_{Val}$. This order is relevant however for defining the transition system $T_{\mathcal{G}}$ (Definition 2.7), so for clarity we may notate it as $T_{\mathcal{G}}[\leq_{Val}]$. Note that the assumption that the order is well-founded is only necessary for justifying the conclusion regarding termination of $T_{\mathcal{G}}$. In particular, the construction of $\mathcal{R}$ is, obviously, completely syntactic, while the justification of its size-change arcs can be put in more general terms as follows.

For a given order $\leq_{Val}$, let $\mathrm{SMO}^{\dagger}[\leq_{Val}]$ denote the simple multiset partial order (as described in Section 4.2) parameterized by the order $\leq_{Val}$ for set elements.

Definition 5.1. For given orders $\leq_{Val}$ and $\leq_B$ (with carrier sets Val and $[0, B]$, respectively), let $(Val \otimes^n B)$ denote the set of tuples $\langle v_1, v_2, \dots \rangle$ of even length, where every odd position is a non-empty multiset over Val, together containing n elements; and every even position is an integer in $[0, B]$. The lexicographic partial order on $(Val \otimes^n B)$, obtained by ordering odd positions with $\mathrm{SMO}^{\dagger}[\leq_{Val}]$ and even positions with $\leq_B$, is denoted by $\leq_{Val} \otimes^n \leq_B$.

The following is a parametrized rereading of Observation 4.6:

Claim 5.2. *For a size-change graph G and a vector $\mathbf{v}$,*

$$((f, \sigma), (g, \sigma')) \in T_{\mathcal{G}}[\leq_{Val}] \implies \sigma'(\mathrm{next}(\mathbf{v}, G)) \leq_{Val} \otimes^n \leq_B \sigma(\mathbf{v}).$$

This shows that the interpretation of $\mathcal{R}$ as size-change graphs works in the general setting.

We now move to transposition. In Section 3.2, we observed that the SCT property of $\mathcal{G}$ is preserved under transposition. But what do the transposed graphs describe? The natural answer is given by the equation

$$T_{\mathcal{G}^t}[(\leq_{Val})^t] = (T_{\mathcal{G}}[\leq_{Val}])^t$$

(a transposed relation is defined in the natural way; $\leq^t$ is the reverse order relation, $\geq$).

Here is a useful lemma concerning transposition.

Lemma 5.3. $((\leq_{Val})^t \otimes^n \leq_B)^t = \leq_{Val} \otimes^n (\leq_B)^t.$

We leave its proof to the reader; note that using the partial version $\mathrm{SMO}^{\dagger}$ helps to avoid the asymmetric definition of SMO for sets of different size.

Theorem 5.4. *Let $\mathcal{G}$ be fan-in free, positive SCT instance, and let $B = m \cdot 2^n \cdot n$. There exists an indexed set $\{S_f\}$, where $S_f \subseteq V_f^B$ for every $f \in F$, such that the function $\rho_f(\sigma) = \mathrm{MAXVAL}_{\sigma} S_f$ is a ranking function for $\mathcal{G}$.*

This ranking function can be effectively constructed given $\mathcal{G}$.

Proof. Observe that this is a version of Theorem 4.11, with MINVAL changed into MAXVAL (this will be justified shortly) and B multiplied by n to compensate for the use of SMO† instead of SMO (see Section 4.2). Now, we describe the construction of ρ.

Given $\mathcal{G}$, construct $\mathcal{G}^t$; note that will be fan-out free. Thus $\mathcal{R}$ can be constructed from it as in the last section, and it is fan-out free and strict. Since the interpretation of $\mathcal{G}^t$ uses the ordering $\leq_{Val}{}^t$, the semantics of $\mathcal{R}$ is given by a transition system

$$T_{\mathcal{R}}[(\leq_{Val})^t \otimes^n \leq_B].$$

Now, $\mathcal{R}^t$, is strict, fan-in free, and interpreted under the order $((\leq_{Val})^t \otimes^n \leq_B)^t$.

Since $\mathcal{R}$ satisfies SCT (as proved in the last section), so does $\mathcal{R}^t$, and by Theorem 3.8 it has a ranking function ρ of the form stated in the current theorem. This function decreases under the ordering $((\leq_{Val})^t \otimes^n \leq_B)^t$.

Since by Lemma 5.3, $((\leq_{Val})^t \otimes^n \leq_B)^t = \leq_{Val} \otimes^n (\leq_B)^t$, we find that ρ descends under the usual ordering of V_f^B, except that the numeric positions are ordered in reverse ($\leq_B{}^t$ instead of $\leq_B$). But this can be easily fixed by exchanging every value v_{2i} by $B - v_{2i}$, so that descent in $(\leq_{Val} \otimes^n \leq_B)$ is obtained. $\qquad\square$

The simplifications considered in the previous section also apply here.

Corollary 5.5. *Every fan-in free, positive SCT instance has a ranking function of form $\rho_f(\sigma) = \mathrm{MAXVAL}_\sigma S_f$ with $S_f \subseteq \widetilde{V}_f^B$ and $|S_f| \leq n!$.*

6. Lower Bounds

Our upper bound on the size of the ranking functions is exponential; more precisely, up to $n!$ vectors under the **min** or **max** sign, for every flow-point. What is the true complexity? In this section we provide (mostly) explicit lower bounds, that is, lower bounds on the "size" of *any* ranking function for a specific family of SCT instances. In order to prove such a result, it is necessary to make assumptions on the form of the ranking function. We progress through three types of ranking functions that generate vectors of variables and constants, as our constructions do. The first type precludes the use of multisets, the second allows them with a restriction, and the third is the most general.

All our examples are for a CFG with a single node f. We consider a class of ranking functions that can be described by assigning an element $\mathbf{v} \in V_f^B$ to any total order τ on $Var(f)$, such that if the values of variables in state s satisfy τ, then $\rho_f(s)$ is given by $\mathbf{v}$ (we can ignore the value of the function in the case that some variables tie). The value of B is left unspecified. Note that the constructions in this paper are of this kind, since if the order of variable values is known, the lexicographic or multiset minima and maxima can be deduced. We call such ranking functions VSO (for Vectors Selected by Order).

Definition 6.1. Let $s = (f, \sigma)$ be a state, and $\pi \in S_n$ an n-element permutation. We say that a state s has order π whenever $\pi i < \pi j \iff s(x_i) < s(x_j)$ for all i, j. If the values of variables in s are distinct, π is unique and we denote it by $Order(s)$.

Definition 6.2. Let $\rho(f, \sigma)$ (or $\rho_f(\sigma)$) be a function over states with co-domain V_f^B. Such a function is called a *VSO function* if there is, for each f, a function $\rho_f^* : S_n \to V_f^B$ such that, for all $s = (f, \sigma)$ where the values of all variables are distinct, we have $\rho_f(\sigma) = s(\rho_f^*(Order(s)))$ (recall that $s(\mathbf{v})$ is the value of vector $\mathbf{v}$ in state s).

We knowingly disregard the states where variable values are not distinct.

The lower bounds in this section apply to the number of distinct vectors in the image set of ρ_f^*, i.e. to the size of the set $\rho_f^*(S_n)$. Thus they apply to the number of vectors under the min or max operators in functions expressed as in Theorems 4.11 and 5.4. All our constructions are fan-out free instances, and the lower bounds are very close to the upper bounds we had for this class.

We believe that in a certain sense, a ranking function generated by a general SCT-based construction (i.e., not using any other information, say about *Val*) *has* to be a VSO function (or representable as one) since all that is assumed of the data *Val* is that they are ordered, so decisions can only be based on order; and variable values can be put into tuples or multisets, but not otherwise used in expressions. We have not formalized this intuition, however.

6.1. **Preliminaries.** The set $Var(f)$ is written in the following examples as $\{x_0, x_1, \ldots, x_n\}$ (so there are actually $n+1$ variables). States may be written in the form $[x_0 \mapsto v_0, \ldots, x_n \mapsto v_n]$. In our examples, values are non-negative integers, and x_0 is always largest. We also assume that all variable values are different. Therefore, the ordering of variable values in a state will be described by a permutation on $[1, \ldots, n]$, reflecting the ordering among $x_1, \ldots, x_n$.

We denote by I the identity on $[1, \ldots, n]$ and by ξ_{ij} the permutation exchanging i and j. Composition of permutations is defined by the rule $(\pi_1 \pi_2)i = \pi_1(\pi_2 i)$. For a state s, $s\pi$ denote the result of the action of π on the state s, defined by $(s\pi)(x_i) = s(x_{\pi i})$.

6.2. **Simple ranking functions.** We observed (in Section 4) that it is possible to restrict all our constructions to vectors which do not employ multisets (technically, all odd positions are singletons). A ranking function of this type is *simple*. We begin by proving a lower bound for simple ranking functions.

Theorem 6.3. *Let $n > 0$. There is a fan-out free positive SCT instance $\mathcal{G}$ with a single flow-point, $n + 1$ variables and n size-change graphs such that any simple VSO ranking function ρ_f for $\mathcal{G}$ satisfies $|\rho_f^*(S_n)| \geq n!$.*

Observe that this lower bound matches almost exactly the upper bound of Theorem 4.17.

The proof of the theorem breaks into the following parts: (1) construction of the SCT instance; (2) proof that SCT is satisfied; (3) proof of the lower bound.

Construction 6.1: SCT instance $\mathcal{G}$. Let $\mathcal{G} = \{G_1, \ldots, G_n\}$, where for $k = 1, \ldots, n - 1$,

$$G_k = \{x_k \to x_{k+1}, x_{k+1} \to x_k\} \cup \{x_i \to x_i \mid i \neq k, k+1\} \cup \{x_0 \xrightarrow{\downarrow} x_0\} \quad \text{and}$$
$$G_n = \{x_i \to x_i \mid 1 \leq i < n\} \cup \{x_n \xrightarrow{\downarrow} x_n\}.$$

(See Figure 9 for an illustration.)

Observe that any contiguous sequence of the graphs $G_1, \ldots, G_{n-1}$ represents (or "effects") some permutation π on $\{x_1, \ldots, x_n\}$, in the sense that every x_i is carried by a (non-descending) thread to $x_{\pi i}$. Henceforth, we call this "a permutation multipath" M_π. Note that for every permutation $\pi \in S_n$, there is a multipath effecting it, less than n^2 long, according to well-known ways of composing any permutation of exchanges.
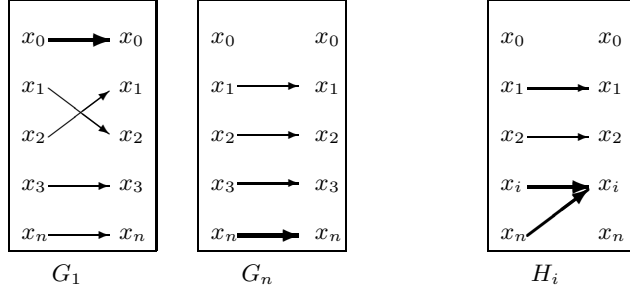
Figure 9: A partial view of two graphs from Construction 6.1 and one from Const. 6.2.

Lemma 6.4. $\mathcal{G}$ *satisfies SCT.*

Proof. In any infinite $\mathcal{G}$-multipath that does not contain G_n (or where G_n occurs a finite number of times), there is infinite descent at x_0. In a multipath that includes an infinity of G_n's, the thread at 0 is lost but each of $x_1, \ldots, x_n$ begins a separate infinite thread. At each occurrence of G_n, one of these threads descends, so there is at least one thread of infinite descent. $\square$

Lemma 6.5. *Let ρ_f be a simple ranking function for $\mathcal{G}$ and $\mathbf{v} = \rho_f^*(\pi)$ for some permutation π. Then x_0 is the last variable that appears in $\mathbf{v}$.*

Proof. Consider a state s with $Order(s) = \pi$, so $\rho_f(s) = s(\mathbf{v})$. Let $i > 0$. Let s' be identical to s except that x_i decreased and x_0 increased, while the relative order of variables remains π, so $\rho_f(s')$ is also given by $\mathbf{v}$. Note that $M_{\xi_{in}} G_n M_{\xi_{ni}} \models s \mapsto s'$. Thus, we must have $\rho_f(s) > \rho_f(s')$, so x_i must appear in $\mathbf{v}$ before x_0. Since this conclusion holds for every $i > 0$, it follows that x_0 must be last. $\square$

For any $\mathbf{v}$, let $\mathrm{LIST}(\mathbf{v})$ denote the permutation that describes the placement of variables in $\mathbf{v}$ (excluding x_0 which we know to be last). That is, $\mathrm{LIST}(\mathbf{v})i = j$ indicates that the ith variable occuring in $\mathbf{v}$ is x_j. For a permutation π, we use the following abbreviations: $\mathbf{v}_\pi$ is $\rho_f^*(\pi)$; L_π is $\mathrm{LIST}(\mathbf{v}_\pi)$.

Lemma 6.6. *Let ρ_f be a simple ranking function for $\mathcal{G}$ and $\pi, \tau \in S_n$. Then $\tau L_\tau = \pi L_\pi$. Moreover, vectors $\mathbf{v}_\tau$ and $\mathbf{v}_\pi$ have the same constants in the even positions below position $2n$.*

Proof. Assume the contrary. Consider the first position that violates the lemma. Suppose first that it is an odd position $2i - 1$, containing a variable. Thus, letting $k = L_\tau i$ and $j = L_\pi i$, we have $\tau k \neq \pi j$. Assume for the rest of the proof that $\tau k > \pi j$ (otherwise, exchange τ and π).

Let $s = [x_j \mapsto \pi j$ for $1 \leq j \leq n, x_0 \mapsto n + n^2]$. Note that $Order(s) = \pi$. Thus, $\rho_f(s) = s(\mathbf{v}_\pi)$. Let $s' = [x_k \mapsto \tau k$ for $1 \leq k \leq n, x_0 \mapsto n + 1]$. Note that $Order(s') = \tau$. Thys, $\rho_f(s') = s'(\mathbf{v}_\tau)$. It should be easy to see that $M_{\tau^{-1}\pi} \models s \mapsto s'$.

For all positions up to $2i - 1$, the contents of $\rho_f(s)$ and $\rho_f(s')$ are the same. In position $2i - 1$, $\rho_f(s')$ has $s'(x_k) = \tau k$, while $\rho_f(s)$ has $s(x_j) = \pi j$. Now, By our assumption, $\tau k > \pi j$, so $s'(x_k) > s(x_j)$. We conclude that lexicographic descent fails, which contradicts ρ_f being a ranking function.

Next, assume that the first position that violates the lemma is an even position $2i < 2n$, containing a constant. The refutation is very similar. $\square$

The proof of Theorem 6.3 is now concluded since the last lemma implies that a distinct vector corresponds to every permutation.

6.3. **Simply-ordered multisets.** Now we move to ranking functions with multisets of any size. In Section 2.5 we described forms of multiset ordering; our ranking functions are constructed so that the change in every multiset value across a transition alway agrees with what we called *the simple multiset order* or SMO. This means that the relation between multisets of the same size must be expressed by a 1–1 correspondence of the elements. This subsection gives a lower bound under this restriction.

Theorem 6.7. *Let $n > 0$. There is a fan-out free positive SCT instance $\mathcal{H}$ with a single flow-point, $n + 1$ variables and $2n - 1$ size-change graphs, such that any SMO-descending VSO ranking function ρ_f for $\mathcal{H}$ satisfies $|\rho_f^*(S_n)| \geq n!$.*

Observe that this lower bound, too, closely matches the upper bound of Theorem 4.11, though a larger class of functions is considered. The proof proceeds through the same stages as the last one.

Construction 6.2: SCT instance $\mathcal{H}$. Let $\mathcal{H} = \{G_1, \ldots, G_{n-1}, H_1, \ldots, H_n\}$, where the G_k graphs are identical to those of Construction 6.1, namely

$$G_k = \{x_k \to x_{k+1}, x_{k+1} \to x_k\} \cup \{x_i \to x_i \mid i \neq k, k+1\} \cup \{x_0 \xrightarrow{\downarrow} x_0\}$$

while for $i = 1, \ldots, n$,

$$H_i = \{x_j \to x_j \mid 0 < j < i\} \cup \{x_j \xrightarrow{\downarrow} x_i \mid j \geq i\}.$$

Thus, $\mathcal{H}$ contains $\mathcal{G}$ (note that H_n is the same as G_n of $\mathcal{G}$).

Lemma 6.8. *$\mathcal{H}$ satisfies SCT.*

Proof. In any infinite $\mathcal{H}$-multipath that does not contain H-graphs (or contains a finite number of them), there is infinite descent at x_0. In a multipath that includes an infinity of H's, the set $\{x_1, \ldots, x_n\}$ descends infinitely in dual multiset order, ensuring termination (the restriction to SMO in Theorem 6.7 only applies to our ranking function, not to this proof!). □

Lemma 6.9. *Let ρ_f be a SMO-descending VSO ranking function for $\mathcal{H}$ and let $\mathbf{v}_\pi = \rho_f^*(\pi)$ for some permutation π. Then x_0 only appears in $\mathbf{v}_\pi$'s last position.*

Proof. Essentially the same as for Lemma 6.5. Note that since x_0 increased from s to s', we cannot obtain simple multiset descent in any set of variables that includes x_0. Therefore, x_i must appear strictly before x_0, not even in the same set. □
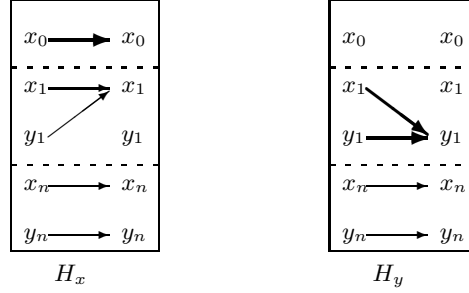
Figure 10: A partial view of two graphs from Construction 6.3.

Lemma 6.10. *Let ρ_f, π and $\mathbf{v}_\pi$ as above. Then every odd position of $\mathbf{v}_\pi$ is a singleton.*

Proof. The proof is by contradiction, as in Lemma 6.6. Consider the first position that violates the lemma for some $\pi \in S_n$. Suppose that it is position $2i - 1$. Let A be this element of $\mathbf{v}_\pi$, that is, the ith set-valued element. Let x_a be the variable in A such that $b = \pi a$ is smallest.

Let $s = [x_j \mapsto 2\pi j$ for $1 \le j \le n, x_0 \mapsto n + n^2]$. Note that $Order(s) = \pi$. Thus, $\rho_f(s) = s(\mathbf{v}_\pi)$. Let $s_1 = [x_j \mapsto 2j$ for $1 \le j \le n, x_0 \mapsto n + 1]$. Note that $Order(s) = I$, and that $M_\pi \models s \mapsto s_1$. Let

$$s_2 = [x_j \mapsto 2j \text{ for } j < b, \ x_b \mapsto 2b - 1, \ x_j \mapsto 2n + j \text{ for } j > b, x_0 \mapsto n + n^2].$$

Note that $H_b \models s_1 \mapsto s_2$, and $Order(s_2)$ is I. Let $s_3 = (s_2\pi)[x_0 \leftarrow n^2 + 1]$, so that $M_{\pi^{-1}} \models s_2 \mapsto s_3$. We have $Order(s_3) = \pi$ again.

For all positions up to $2i - 1$, the variables and constants in $\rho_f(s_3)$ have at least their value as in $\rho_f(s)$. In position $2i - 1$, we have a set that includes the variable x_a, and, if $|A| > 1$, also other variables of greater indices. In s_3, those other variables have values larger than in s. Thus, if lexicographic descent (under SMO) is to be maintained, we must conclude that $|A| = 1$. □

Lemma 6.11. *Let ρ_f be a restricted ranking function for $\mathcal{H}$ and let $\pi, \tau \in S_n$. Let $\mathbf{v}_\pi = \rho_f^*(\pi)$, and likewise $\mathbf{v}_\tau = \rho_f^*(\tau)$. Then $\tau L_\tau = \pi L_\pi$. In addition, vectors v_τ and v_π have the same constants in the even positions before position $2n$.*

Proof. Given the last lemma, the situation is identical to that in Lemma 6.6 and the same proof holds. □

The proof of Theorem 6.7 is now concluded since the lemma implies that a distinct vector corresponds to every permutation.

6.4. **Dual-ordered multisets.** We now extend the allowable range of ranking functions further by allowing a stronger type of multiset ordering to be used. Since our example consists of fan-out free graphs, it is easy to conclude that among the two orders described in Section 2.5, it is *dual multiset order* (DMO) which is promising to be useful. So, we consider VSO ranking functions in which multisets are compared by DMO.

Theorem 6.12. *There is a fan-out free positive SCT instance $\mathcal{K}$ with a single flow-point, $2n + 1$ variables and $n + 1$ size-change graphs, such that any VSO ranking function ρ_f for $\mathcal{K}$ (with DMO descent) must use at least 2^n different vectors.*

Construction 6.3: SCT instance $\mathcal{K}$. The variables in this example are named $x_0, \ldots, x_n$ and $y_1, \ldots, y_n$.

Let $\mathcal{K} = \{G_1, \ldots, G_{n-1}, H_x, H_y\}$, where the G_k graphs are permutation graphs operating on pairs (x_i, y_i):

$$
\begin{aligned}
G_k \quad = \quad & \{x_k \to x_{k+1}, x_{k+1} \to x_k, y_k \to y_{k+1}, y_{k+1} \to y_k\} \\
& \cup \{x_i \to x_i, y_i \to y_i \mid i \neq k, k+1\} \\
& \cup \{x_0 \overset{\downarrow}{\to} x_0\}
\end{aligned}
$$

Graphs H_x, H_y operate specially on the first pair:

$$
\begin{aligned}
H_x \quad = \quad & \{x_1 \to x_1, y_1 \to x_1\} \cup \{x_i \to x_i, y_i \to y_i \mid i > 1\} \cup \{x_0 \overset{\downarrow}{\to} x_0\}; \\
H_y \quad = \quad & \{x_1 \overset{\downarrow}{\to} y_1, y_1 \overset{\downarrow}{\to} y_1\} \cup \{x_i \to x_i, y_i \to y_i \mid i > 1\}
\end{aligned}
$$

Lemma 6.13. *$\mathcal{K}$ satisfies SCT.*

Proof. Variable x_0 guarantees descent, except for multipaths that contain H_y. It suffices to prove that for each such finite multipath $M = M_1 H_y M_2$, M^ω includes an infinitely-descending thread[1]. Note that M has this property if and only if $M' = M_2 M_1 H_y$ does. Now, M' has at least two descending threads that end at y_1, beginning at x_i and y_i for some i. If $i = 1$, the thread from y_1 is multiplied infinitely in $(M')^\omega$. If $i \neq 1$, observe that the effect of M' on the pairs is to permute them, so there is a k (the order of the permutation) such that $(M')^k$ has a y_1 to y_1 thread as in the simple case, and we have the same conclusion regarding $(M')^\omega$. $\qquad\square$

In proving the lower bound on the number of vectors we use tools similar to those of the previous proofs. Now, however, we restrict the orderings of variables on which we focus in the proof, so that the pairs are kept in increasing order, i.e.,

$$\max(x_1, y_1) < \min(x_2, y_2) < \max(x_2, y_2) < \min(x_3, y_3) < \cdots$$

and, as usual, x_0 is larger than the rest. The relative order among the elements of each pair may change. The ordering of pairs is described by a function $\alpha : \{1, \ldots, n\} \to \{\mathtt{x}, \mathtt{y}\}$ that indicates, for every pair, which variable has the smaller value. Let S_n' be the set of such functions. Thus the order of values in a given state is described by an element of S_n', and we use S_n' as the domain of ρ_f^*.

The basic properties of a ranking function ρ_f for this instance follow the pattern of previous examples, so we omit a detailed proof:

(1) x_0 appears last.
(2) The constants in the even positions below $2n$ coincide for all vectors returned by ρ_f^*.
(3) For every odd position, the size of the set is the same in all vectors returned by ρ_f^*.

The property specific to this construction is as follows.

Lemma 6.14. *Let $\alpha \in S_n'$. Let $\mathbf{v}_\alpha = \rho_f^*(\alpha)$. Then from every pair $\{x_i, y_i\}$, only the element selected by α, that is, the variable of smaller value, is present in $\mathbf{v}_\alpha$.*

Proof. The proof is by contradiction, as usual. Consider the first position that violates the lemma. Suppose that it is an odd position $2i - 1$, containing a set A.

Case 1: $x_j \in A$, but $\alpha(j) = \mathtt{y}$. Suppose first, for simplicity, that $j = 1$. Consider a state s described by α, and a state s' where the value of x_1 is greater than its value in s, the value of y_1 is smaller than in s, other values do not change, and neither does the relative

---

[1] This sufficiency claim is implicit in the proof of Theorem 4 in [10].

order among all variables. Thus, the order at s' is also given by α. Note that $H_y \models s \mapsto s'$. The multiset of values of A increases from s to s', unless A includes y_1. So, we conclude that both x_1 and y_1 are in A. Now, consider H_x; we can easily have a transition that increases the multiset value of A and consequently a lexicographic increase in the value of ρ_f.

The case where $j > 1$ is proved by the same argument, using $M_{\xi_{j1}} H_y M_{\xi_{j1}}$ (here $M_{\xi_{j1}}$ is a multipath that exchanges the pair (x_j, y_j) with (x_1, y_1)).

Case 2: A includes y_j, while $\alpha(j) = \mathbf{x}$. Again, for simplicity, assume $j = 1$ (otherwise use $M_{\xi_{j1}}$ as above). Consider a state s described by α, and a state s' where the value of y_1 is greater than its value in s, other values (except for x_0) do not change, and neither does the order. This agrees with H_y. Now, $\rho_f(s') = s'(\mathbf{v}_\alpha)$; the multiset of values of A increases from s to s', a contradiction to correctness of ρ_f. $\quad\square$

The proof of Theorem 6.12 is now concluded since the lemma implies that a distinct vector corresponds to each of the 2^n orderings in S'_n.

Remark 1: 2^n is tight for the example; a possible ranking function for $\mathcal{K}$ is one that returns the minimum over a set of vectors, all of length 3, constructed as follows[2]: The first position contains a set of n variables—one from every pair. All 2^n such sets are present. For a vector that begins with set S, the next position contains the number of y's in S. The third position is $\{x_0\}$.

Proof: to prove lexicographic descent, the non-trivial case is a transition that does not decrease the minimum among the multiset values in the first position (clearly, it never increases). Suppose that in such a transition, the old minimum vector was $\mathbf{v}_1 = \langle S_1, k_1, \dots \rangle$ and the new one is $\mathbf{v}_2 = \langle S_2, k_2, \dots \rangle$. If $k_2 < k_1$, we have descent. So, assume $k_2 \geq k_1$, and consider what transition was taken.

- If it is H_x, S_2 can only differ from S_1 if the minimum among $\{x_1, y_1\}$ moved from y_1 to x_1 (the new value of y_1 being at least as large). But then the number of y's in S_2 is smaller. If S_2 and S_1 are the same, we have lexicographic descent because x_0 decreases.
- If it is H_y, S_2 must differ in value from S_1. In fact, there is dual multiset descent in $\{x_1, y_1\}$.
- If it is G_i, given that the multiset does not decrease, S_2 has to be the same set as S_1. So $k_2 = k_1$, and x_0 decreases. $\quad\square$

Remark 2: The lower bound dropped from $2^{\Theta(n \log n)}$ to $2^{\Theta(n)}$. What is the true complexity? It's an intriguing possibility that the use of multisets (beyond SMO) might decrease the ranking-function complexity (for a single flow-point) to $2^{O(n)}$.

6.5. On the program size of free-form ranking functions.
A "free form" ranking function is described by a program; consider, for instance, a program that sorts the variables and outputs the sorted list. The size of a program expressing this function can be much smaller than the number of vectors in its image ($n!$). Giving an explicit lower bound on the size of general programs seems quite hard. Instead, [2] argues that polynomial ranking functions are very unlikely to exist, under standard complexity-theoretic assumptions. This is shown for all-strict SCT, using a proof that the decision problem (does an all-strict SCT instance terminate?) is PSPACE-hard. We note that the decision problems for fan-in free (or fan-out free) SCT are also PSPACE-hard ([10] gives the proof for fan-in free graphs and

---

[2]Technically, we need to add a dummy fourth element to match the definition of V^f.

the fan-out free case follows easily). Thus, the same complexity-theoretic conclusion applies to these sub-problems as well.

7. The Ordinal Height of Ranking Functions

The fact that *existence* of a global ranking function follows from a termination proof of the local type is immediate from the fact that every terminating program has a global ranking function. However what can be deduced from the form or number of local ranking functions on the form or complexity of a global one is a difficult question. A recent paper by Blass and Gurevich [5] settles the question of the *ordinal height* of the global ranking function.

- Consider any transitive relation R (in our setting, this will be the transitive closure of the transition relation of the subject program, or the SCT transition system as described in Definition 2.7). The goal of the "termination proof" is to establish that R is well-founded.
- The goal is achieved by describing a finite set $U_1, \ldots, U_m$ of well-founded relations such that $R \subset U_1 \cup \cdots \cup U_m$.
- Blass and Gurevich give a general upper bound on the ordinal height $|R|$ of R in terms of the ordinals $\alpha_i = |U_i|$.

In our setting, the covering relations U_i will include two types of relations:

(1) For all pairs (f, g) of distinct flow-points, there is a relation that includes all pairs of states $((f, \sigma), (g, \sigma'))$. This relation is well-founded because there are actually no chains in it of length greater than one. Removing these parts from R leaves only the part that describes cycles, i.e., pairs $((f, \sigma), (f, \sigma'))$.

(2) The second group of relations U_i covers all cycles.

Obviously, the latter group is the interesting one for termination proofs as well as for the ordinal-height question, and to simplify the present discussion, we can restrict our attention to the case $|F| = 1$, so that only the second kind is left.

In order to get the best bound out of the Blass-Gurevich theorem, we want to cover R in the most economical way. To this end we use the following theorem, a rewording of [6, Corollary 1].

Theorem 7.1 (Codish-Lagoon-Stuckey). *Size-change graph* $G : f \to f$ *over parameters* $x_1, \ldots, x_n$ *satisfies SCT if and only if*

$$G \models (f, \sigma) \mapsto (g, \sigma') \implies \bigvee_{1 \leq i \leq n} \sigma(x_i) > \sigma'(x_i).$$

Simply put, this means that the set of n relations $\sigma(x_i) > \sigma'(x_i)$ covers the transition relation of G. Therefore, the Blass-Gurevich result applies with exactly n relations U_i, all isomorphic to *Val*. In this simple case, Blass and Gurevich's bound becomes α^n, where α is the ordinal height of *Val* (e.g., ω when *Val* is the natural numbers, the most usual case).

Our explicit constructions match this bound, at least for typical domains. Consider the construction where the range of the ranking function is described by vectors of length $2n$, alternating parameters and integers bounded by B. The order type of this range is $(B\alpha)^n$. Whenever α is a multiple of ω, this is exactly α^n.

While the ordinal bound matches, our results in this work are not a corollary of Blass and Gurevich's result. Note that in the above argument we covered all SCT instances with

the same set of n local functions. Obviously, the identity of this set of functions cannot reveal any specific structure of a given instance, as does an expression for a ranking function.

8. Conclusion

While deducing ranking expressions from size-change graphs has already been shown possible before this work, the constructions in this paper are simpler and more transparent than previously known. They improve the upper bound on the size of the ranking expression and in fact achieve optimality, in a certain sense.

The constructions employ reductions of SCT instances to instances of a subclass of SCT, and applies SCT to data of composite types (tuples and sets). We feel that this technique is interesting in itself.

To argue for optimality, we have introduced a class of expressions that (in our opinion) captures all possible ranking functions for general SCT, and a complexity measure (number of different vectors in the image) under which we are able to prove lower bounds.

Several theoretical problems remain. For example:

- For fan-in/-out free graphs, will the use of multisets and multiset ordering allow ranking functions of size 2^n to be constructed? Or is our lower bound loose?
- What is the complexity of ranking functions for general SCT, and how to generate them?
  We remark that our construction relied on fan-out freedom in the construction of the instance $\mathcal{R}$ (Section 4.1). Furthermore, it is not hard to verify that the forms of ranking functions given by our constructions do not suffice for certain SCT instances which are not fan-in or fan-out free. The strict SCT instance shown on Page 9 is such an example. On the other hand, practically, there is evidence that fan-in free graphs are common. For example, in analysing a benchmark of SCT instances derived from Prolog programs [4], we discovered that fan-in occurred rarely once size-change graphs have been "cleaned up" by removing arcs unnecessary for the termination proof.
- How can the construction algorithm given in this paper be improved? Note that as presented, it may require doubly exponential time and space, despite the fact that the size of the final result is bounded by $m \cdot 2^{O}(n \log n)$.

Practically, the choice of an algorithm for ranking-function construction and its usage are also challenging. It is well known that even algorithms that are worst-case exponential sometimes work sufficiently well in many practical cases. It is quite possible that human-written programs will not require ranking functions of high complexity. In this work, these practical questions have not been studied, as our goal was to examine the theoretical problem first. Recent work by Ben-Amram and Codish [3] proposes to use a different class of ranking functions which does not cover all SCT instances, but has polynomial expression size; and it turns out to suffice for the benchmark that was tried. The compact representation of the ranking functions relies on the use of sets of variables inside tuples, an insight gained from the work described in this paper.

Even if a ranking-function construction is provided, the practical goals mentioned—certified termination, proof carrying code and execution time analysis—all require additional research and implementation work for their realization.

References

[1] Amir M. Ben-Amram. Size-change termination with difference constraints. *ACM Transactions on Programming Languages and Systems*, 2007. to appear.

[2] Amir M. Ben-Amram. A complexity tradeoff in ranking-function termination proofs. *Acta Informatica*, 46(1):57–72, February 2009.

[3] Amir M. Ben-Amram and Michael Codish. A SAT-based approach to size change termination with global ranking functions. In C.R. Ramakrishnan and Jakob Rehof, editors, *14th Intl. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 5028 of *LNCS*, pages 46–55. Springer, 2008.

[4] Amir M. Ben-Amram and Chin Soon Lee. Size-change analysis in polynomial time. *ACM Transactions on Programming Languages and Systems*, 29(1), 2007.

[5] Andreas Blass and Yuri Gurevich. Program termination and well partial orderings. *ACM Transactions on Computational Logic*, 9(3):1–26, 2008.

[6] Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Testing for termination with monotonicity constraints. In Maurizio Gabbrielli and Gopal Gupta, editors, *Logic Programming, 21st International Conference, ICLP 2005*, volume 3668 of *Lecture Notes in Computer Science*, pages 326–340. Springer, 2005.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[8] Alexander Krauss. Certified size-change termination. In Frank Pfenning, editor, *11th International Conference on Automated Deduction (CADE)*, volume 4603 of *LNAI*, pages 460–475. Springer-Verlag, July 2007.

[9] Chin Soon Lee. Ranking functions for size-change termination. *ACM Transactions on Programming Languages and Systems*, 2009. to appear.

[10] Chin Soon Lee, Neil D. Jones, and Amir M. Ben-Amram. The size-change principle for program termination. In *Proceedings of the Twenty-Eigth ACM Symposium on Principles of Programming Languages, January 2001*, volume 28, pages 81–92. ACM press, January 2001.

[11] Andreas Podelski and Andrey Rybalchenko. Transition invariants. In Harald Ganzinger, editor, *LICS'04: Logic in Computer Science*, pages 32–41. IEEE Computer Society, 2004.

[12] Yehoshua Sagiv. A termination test for logic programs. In Vijay Saraswat and Kazunori Ueda, editors, *Logic Programming, Proceedings of the 1991 International Symposium, San Diego, California, USA*, pages 518–532. MIT Press, 1991.

[13] René Thiemann and Jürgen Giesl. The size-change principle and dependency pairs for termination of term rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 16(4):229–270, September 2005.